TOPICS TO LEARN

### Created By

Raazi Muahmmed

[Linkedin - Raazi Muhammed](https://www.linkedin.com/in/raazimuhammed/)

### Contributors

Ajmal

[Linkedin - Ajmal Jaleel](https://www.linkedin.com/in/ajmal-jaleel-1267b9275/)

### 

Hafsana

[Linkedin - Hafsana S](https://www.linkedin.com/in/hafsana-s-615103287/)

### 

Akarsh

[Linkedin - Akarsh G Kumar](https://www.linkedin.com/in/akarshgkumar/)



Rabeeh

[Linkedin - Rabeeh pk](http://www.linkedin.com/in/rabeeh-pk)

### 

### TABLE OF CONTENTS

[HTML & CSS 2](#_4mo29mozu2nj)

[JavaScript 3](#_45pv78mg7x7q)

[Node.js, Express 7](#_rz6bzr6ymcr2)

[MongoDB 11](#_oe1pq84mq4jz)

[React 16](#_cp4rer7rom82)

[Redux 20](#_iyt0v0hl3vfz)

[React Native 22](#_iph2xmlozcp4)

[DSA 23](#_79uxddpacbd4)

[Hosting 28](#_6bwp1fh5rcen)

[System Design 29](#_c7xqmpawun7a)

[Git 30](#_lbi9vxeje4w)

[SQL: Postgres 32](#_qzrywgqwzksb)

[Microservice 35](#_y6id2m1um17)

[Docker 38](#_y6vzu4qbwcyk)

[Kubernetes 40](#_nvqopd9ka1zy)

[Message Broker 43](#_dcb6k40n1vs)

[TypeScript 45](#_xtao36eo0lan)

[Next.js 46](#_3spkphontn4t)

[NestJS 48](#_oiijv1p0znw1)

[Clean Code 50](#_pr8p8ft4sin9)

[Python 51](#_9jqkdsm4z04t)

[Others 54](#_ixvosomi6z4e)

[For detailed explanations](https://github.com/muhammadfarhankt/MERN-Stack-Developer-Roadmap)

Since people have been marking comments as resolved, I have set the docs to view only. If you need to see the comment please request permission to edit

# HTML & CSS

## HTML

1. **Basics**
2. **Block element and inline element**
3. Element
   1. Void elements
   2. Container Element
4. Attributes
   1. boolean attributes
   2. lang attribute
5. Nesting
6. <!DOCTYPE html>
7. **head**
   1. **<meta>**
   2. <meta charset="utf-8">
   3. Adding an author and description

### VS

1. h1 vs title in head
2. <em> vs <i>
3. <b> vs <strong>

### GOOD TO KNOW

1. Whitespace
2. entity references
   1. < &lt;
   2. > &gt;
   3. " &quot;
3. Open Graph Data

## CSS

1. Anatomy of CSS ruleset
2. Selecters
   1. Element
   2. Id, Class
   3. Attribute
   4. Pseudo
3. Box model

# JavaScript

## DOM

* 1. querySelector
  2. textContent
  3. addEventListener
  4. Order of Parsing

1. **event Propagation**
   1. event Bubbling
   2. event Capturing/ Trickling
   3. how to add both on program
2. event.stopPropagation();
3. inst
   1. e.target
      1. id
      2. tagName
      3. pros and cons

## Architecture

* 1. Execution context
     1. variable environment (memory)
     2. Thread of execution (code)
     3. - global & local execution context
     4. - phases
        1. Memory allocation
        2. Code execution
  2. Synchronous single threaded app
  3. Call stack
  4. Proxy
     1. Proxy traps
     2. Reflect
     3. proxy vs reflect
  5. **Event loop**
     1. Callback queue/ task queue
     2. Microtask queue
        1. mutation observer
     3. Starvation
     4. Memory Heap
  6. Just In Time Compilation
  7. Interpreter vs Compiler
  8. Abstract Syntax Tree
  9. Concurrency model

## Theory

1. Data types
   1. wrapper objects
   2. 0 vs new Number(0)
   3. **Numbers**
      1. 1\_000\_000
      2. 1e9, 1e-6
      3. Hex, binary and octal numbers
      4. toString(base)
      5. Math.trunc
2. Operators
3. enum
   1. how to get enum in javascript
4. **Function**
   1. Function Statement
   2. Function Expression
   3. Function Declaration
   4. Anonymous function
   5. Named Function Expression
   6. Functional Programing
   7. **Higher order function**
   8. First class function
   9. **Decorator function**
      1. use
      2. - count no of function call
      3. - valid data of params
   10. **Pure function**
       1. pros and cons
       2. rules
       3. pure vs impure
   11. IIFE
       1. pros
5. Advantages and disadvantages of JS
6. **Set Map Flat**
   1. set
      1. add, delete, has, clear, kyes, values, entries
      2. <setName>.size
   2. map
      1. get, set, has, delete, clear, keys, values, entries, forEach
      2. iterating
   3. object vs map
   4. weekSet()
      1. features
   5. weekMap()
      1. features
      2. key is private
   6. Week set and map summary
   7. falt()
   8. flatMap()
   9. reduceRight()
   10. copyWithin()
7. **Operators**
   1. Nullish coalescing operator
   2. Optional chaining
   3. || vs ??
   4. Ternary operator
   5. Type Operators
   6. **Unary operators**
      1. delete
      2. typeof
      3. !, ++, -, +
   7. **Bitwise Operators**
      1. bitwise OR
      2. bitwise AND
      3. uses
8. **Scope**
   1. Global scope
   2. Module scope
   3. Function scope
   4. Lexical scope
   5. Block scope
9. Shadowing & Illegal shadowing
10. **Prototype**
11. Types of error
    1. syntax, logic
12. **Closure**
    1. Disadvantage
    2. Uses
    3. lexical scope vs closure
    4. IIFE
13. **Garbage collection**
    1. How does it work?
    2. mark-and-sweep
    3. reachability
    4. **Optimizations**
       1. - Generational
       2. collection
       3. - Incremental collection
       4. - Idle-time collection
14. **Hoisting**
    1. TDZ let, const vs var
    2. Function vs arrow function
15. **Call Apply Bind**
    1. function borrowing
    2. call vs apply vs bind
    3. polyfills
16. transpiler
    1. Babel.
    2. webpack
17. polyfills vs transpiler
18. This Keyword
19. **String Methods**
    1. Length, toUpperCase, LowerCase, Trim, Pad, charAt, Split, Concat, substring, indexOf, lastIndexOf, localeCompare
20. **Array Methods**
    1. Map, Filter, Reduce, Find, Sort, Foreach, Push, Pop, Shift, Unshift, Slice, Splice, concat, indexOf, lastIndexOf, forEach, split, join, reduceRight, iArray, fill, copy, flat
    2. spare array, jagged array, hols in array
    3. copy within
    4. typed arrays
21. **Object Methods**
    1. object constructor, literal
    2. deleting field
    3. Computed properties
    4. \_\_proto\_\_
    5. in
    6. Object.assign
    7. structuredClone
    8. \_.cloneDeep(obj)
    9. methods
    10. this keyword
    11. Symbol type
22. Symbol
    1. properties
    2. use
    3. ongo
    4. ​​global symbol registry
    5. for, keyFor, iterator, toPrimitive
23. **Loop**
    1. for
    2. do while vs while
    3. labelled statements
    4. - break
    5. - continue
    6. for…in
    7. for…of
24. **Callback**
    1. callback hell
    2. inversion of control
25. **Promises**
    1. Promise states
    2. Promise chaining
    3. Promise.all
    4. Promise.allSettled
    5. Promise.any
    6. Promise.race
    7. Promise.resolve
    8. Thenable
    9. Finally
    10. Catch
    11. immutable
    12. promisify
    13. pros and cons
26. **Async await**
    1. async always return a promise
    2. error handling in async await
27. **Debouncing & Throttling**
    1. both are used for optimising performance of a web app
    2. by limiting the rate of API calls
28. Spread and Rest Operator
29. DOM, BOM
30. Window Object
31. **ES6 and its features**
    1. Let, Var, Const
    2. Ternary operator
    3. Arrow function
    4. Template literals
    5. Default Parameters
    6. Classes
    7. Modules
    8. Iterators
    9. Object & Array Destructuring
32. **Primitive and non-primitive**
    1. Pass by value and pass by reference
33. Message queue
34. Life
35. Generator
36. **Prototype**
    1. Prototype chain
    2. Prototypal Inheritance
    3. uses?
    4. Circular reference
    5. Object.key
37. **Recursion**
    1. recursive call to function
    2. condition to exit
    3. pros and cons
    4. display the fibonacci sequence
    5. use
38. JavaScript is dynamically types
39. **Currying** 
    1. function inside function
40. **Type Casting**
    1. Implicite (Coercion)
    2. Explicit (Conversion)
41. Microtask queue
42. **Shallow copy vs Deep copy**
    1. primitive vs structural
    2. how to make these copies
    3. pros and cons
    4. Mutable vs Immutable
    5. Object.freeze()
43. TCP/IP
44. DNS
45. **IIFE**
    1. pros and cons
46. **Composition vs Inheritance**
47. Function recursion
48. [Symbol.iterator]
49. Truthy and falsy value
50. Strict mode in JS
51. this substitution

## VS

* 1. label vs func
  2. == and ===
  3. Let, const, var
  4. Synchronous vs asynchronous
  5. While vs do while
  6. Foreach Vs Map
  7. Parameters, Arguments
  8. for in, for of
  9. Undefined, Null
  10. Keywords & Identifiers
  11. Type casting vs Type coercion
  12. textContent vs innerText
  13. identifiers vs variables
  14. defer vs async

## Good to Know

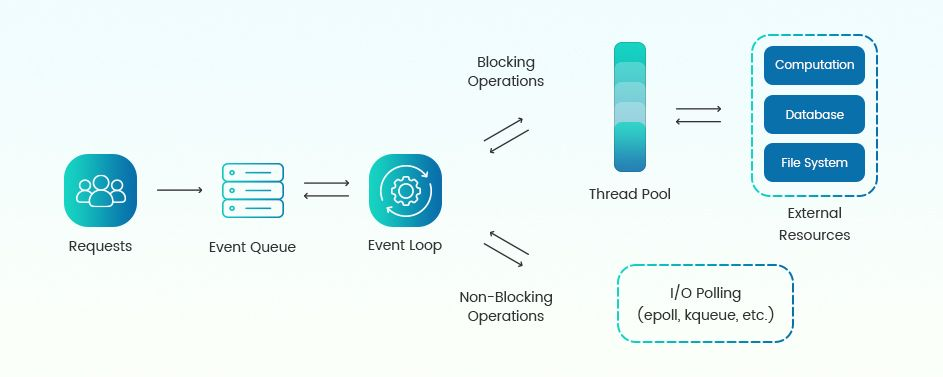
1. interpreted and compiled doe
2. Server-side vs client-side code
3. with in js

# Node.js, Express

## Theory

1. What is Node.js
2. why v8 Engine
3. Advantages & Disadvantages of Node.js
4. How node works
5. libuv
6. Node Module System
7. Concurrency vs parallelism
8. REPL , Cli
   1. \_
9. NPX

### Architecture



1. I/O Polling
2. epoll
3. kquee
4. Thread pool
5. Event queue
6. Event loop
   1. phases
7. External Resources
8. Globals
   1. \_\_dirname
   2. \_\_filename
   3. **Module**
   4. Process
9. **Modules**
   1. **Core Modules.**
   2. local Modules.
   3. Third-party Modules.
   4. module.exports:{}
   5. require
   6. ESM
      1. import and export
10. **NPM**
    1. local and global
    2. npm init
11. npm install or i Nodemon
    1. scripts
       1. start
       2. dev
    2. npm run dev
12. package.json
13. package-lock.json
14. Event loop
15. Event Queue
16. Events
    1. **Events emitter**
       1. on, emit
    2. Http module
17. **Streams**
    1. type of streams
       1. writable, readable, duplex, transform
    2. createReadStream()
    3. readFile vs readFileSync
    4. pipe()
    5. Buffers
    6. Transfer-Encoding: chunked
18. **Cron-job**
    1. \* \* \* \* \*
    2. 1st\* = second
    3. 2nd\* = minute
    4. 3rd\* = hour
    5. 4th\* = day of month
    6. 5th\* = month
    7. 6th\* = day of week
    8. or, range selector
    9. time zone
    10. validation
19. **CORS**
    1. preflight request
       1. header
       2. accept-control-allow-origin: \*
       3. accept-control-allow-methods: \*
20. **Cluster**
21. Multithreading in node.js
    1. require(‘worker\_theads’)
    2. new Worker
22. thread pool
23. worker thread
    1. creating worker,
    2. parent port
24. cluster vs workerthread
25. child process
    1. methods
    2. - fork
    3. - exec
    4. - execFile
    5. - spawn
    6. spawn vs fork
    7. child\_procees.fork() vs cluster.fork()

## HTTP

* 1. https
  2. How does it work?
  3. SSL certificate working
  4. default port
  5. request response cycle
  6. Stateless protocol
     1. Local storage, Sessions and Cookies
  7. Request
     1. General (start line)
        1. method/target/version
     2. header
     3. body
  8. Response
     1. General (start line)
        1. version/statuscode/statustext
     2. header
        1. content type
     3. body
        1. requested resource
  9. **HTTP Methods**
     1. GET
     2. POST
     3. PUT
     4. PATCH
     5. DELETE
     6. HEAD
     7. CONNECT
     8. OPTIONS
     9. TRACE
  10. Idempotent
  11. Safe Methods
  12. User-Agent
  13. Headers
  14. writeHead vs setHead
  15. Status code
      1. 1xx: Informational
      2. 2xx: Success
         1. 200 - Success
         2. 201 - Success and created
      3. 3xx: Redirect
         1. 301: moved to new URL
         2. 304: not changed
      4. 4xx: Client Error
         1. 401: Unauthorised
         2. 402: Payment Required
         3. 403: Forbidden
         4. 404: Page not found
      5. 5xx: Server Error
  16. MIME type
  17. HTTP v2
  18. TCP and IP

1. XSS
2. CSRF
3. MMA
   1. referral header
4. SQL injection
   1. prepared statements

## Express

1. npm install express –save
2. app = express()
   1. get()
      1. status()
      2. send()
      3. sendFile()
   2. post()
      1. express.urlencode()
      2. Form vs JS
   3. put()
   4. patch()
   5. delete()
   6. all()
   7. use()
   8. listen()
3. Static files
   1. public
   2. express.static()
4. **API**
   1. json()
5. **Params, Query String**
6. Route Parameter
7. Query string/url Parameter
8. Path params
9. **MIddleware**
   1. what is middleware
   2. used for what?
   3. req, res, next
   4. next()
   5. app.use in middleware
   6. passing two middleware
   7. **Types of Middleware**
      1. Application-level middleware
      2. Third party middleware
         1. morgan
         2. multer
      3. Router-level middleware
      4. Built-in middleware
      5. Error-handling middleware
         1. err.statusCode
         2. err.message
10. **Routing**
    1. router
    2. express.Router()

### Core Express

* 1. **Session**
     1. i express-session
     2. secret
     3. resave
     4. saveUninitialized
     5. destroy()
  2. **Cookies**
     1. i cookie-parser
  3. Core middleware
  4. Core routing
  5. Build own API
  6. Core views
  7. database integration

### Questions

1. How to send find as response
2. Transaction in node.js

### EJS

* 1. i ejs
  2. server side rendering
  3. view engine
  4. render()
  5. <% %>, <%- %>, <%= %>
  6. partials

1. **Rest API**
   1. RESTful
2. fragment identifier

### VS

1. API vs HTTP
2. API vs SSR
3. HTTP vs HTTPS
4. URIs vs URLs vs URNs
5. Session vs Cookies
6. GET vs POST
7. PUT vs PATCH
8. SSL vs TLS
9. **Build-in Modules (only imp)**
   1. os
   2. path
      1. join()
      2. basename()
      3. resolve()
   3. fs
      1. fs sync
      2. - readFileSync()
      3. - writeFileSync()
      4. - appendFileSync()
      5. - unlinkFileSync()
      6. - statusSync()
      7. - mkdirSync()
         1. recursive: true
      8. **fs async**
      9. - readFile( )
      10. - writeFile()
   4. http
      1. createServer()

# MongoDB

## Theory

1. SQL(relational) v s
2. NoSQL ()
3. What is MongoDB?
4. Run on JS Engine
5. How does mongoDB work?
6. Non-relational Document based
7. Advantage and Disadvantages
8. BSON
9. MongoDB Structure
10. MongoDB architecture
11. JSON vs BSON
12. MongoDB shell
13. CRUD Operations
14. Cursor, Iterate a Cursor
15. Time to Leave
16. Maximum Document Size : 16Mb
17. **Storage engines**
    1. **types**
       1. WiredTiger
       2. ger engine
       3. In-memory engine
       4. MMAPv1
    2. GridFS
    3. Journal
18. **Data types in MongoDB (BSON)**
    1. ObjectId
       1. timestamp
       2. random value
       3. incrementing counter
    2. String
    3. Int, longInt, Double
    4. Array, Object
    5. Boolean
    6. Date
    7. Decimal128
    8. Regex
    9. Javascript
       1. with scope
       2. without scope
    10. MinKey, MaxKey
    11. Binary data
19. Cursor
    1. cursor methods
    2. - toArray
    3. - forEach
    4. cursor.allowPartialResults()
20. **Collection**
    1. db
    2. db.createCollection(collectionName)
    3. show collections
    4. renaming Collection
21. **Documents**
    1. adding new Documents
    2. Nested Documents
       1. advantage
22. **Inserting Document**
23. Insert One and Many
24. what are the additional methods used for inserting
25. **Finding / Querying**
    1. find()+
       1. iterate (it)
       2. pretty()
    2. findOne({ *filter* })
    3. finding In nested Array
       1. “*field*.*field*”
       2. match
       3. exact match
       4. multiple match
    4. Array
       1. finding in specific order
       2. without regard to order
       3. query by array index
       4. query by array length
    5. **Projection**
       1. explicitly include fields
    6. Null, $type: 10, $exists
26. **Filtering**
    1. find( *filter* )
    2. find( *{filter}, {fieldsToGet}* )
27. **Method Chaining**
    1. count()
    2. limit()
    3. sort( 1 or -1 )
    4. skip()
28. **Operators** (denoted by $)
    1. {$gt: number} $gte
    2. $lt, $lte
    3. $eq, $ne
    4. $or $and $not
    5. $in: [1,2,3], $nin: [1,2]
    6. $all
    7. $set, $unset
    8. $addToSet
    9. **$elemMatch**
    10. $slice
    11. $size
    12. $inc: 1, $inc: -1
    13. $pull, $push
    14. $each [ 1, 2 ]
    15. $eq, $ne
    16. $currentDate
    17. $exists
    18. **$expr**
    19. **$cond**
    20. $rename
    21. $min, $max
    22. $mul
    23. $ifNull
    24. $let
    25. **Array Operator**
        1. $push
        2. $each
        3. $pull
        4. $pullAll
        5. $pop
        6. $elemMatch
29. **Deleting**
    1. deleteOne({ *field:value* })
    2. deleteMany()
    3. remove()
    4. delete vs remove
30. **Updating**
    1. updateOne( {*whichObject*} , {$set: {*field: value, field: value*} } )
    2. **Operators**
       1. $set
       2. $unset
       3. $rename
    3. updateMany()
    4. replaceOne()
    5. incrementing & decrementing
    6. adding and remove from array
    7. upsert
    8. update() vs updateOne()
    9. updateOne vs replaceOne
31. **bulkWrite()**
    1. ordered: false
    2. ordered vs unordered
    3. advantages and disadvantages
32. **Commands**
    1. mongosh
    2. db
    3. show dbs
    4. db.stats
33. **Aggregation** 
    1. How does it work
    2. advantages
    3. types of aggregation
    4. distinct
    5. **Aggregate stages**
       1. $addFields
       2. $match
       3. $group
          1. grouping by
          2. -nested field
          3. -multiple field
       4. $sort
       5. $set
       6. $count
       7. - other ways to count
       8. - client and server side counting
       9. $limit, $skip
       10. $merge
       11. $out
       12. $project
       13. $lookup
       14. $unwind
       15. $facet
       16. $fill
       17. $bucket
           1. $bucketAuto
       18. $densify
       19. $redact
       20. $search
       21. allowDiskUse: true
    6. “$name” vs ”name”
    7. **Accumulator Operators**
       1. $sum, $avg, $max, $min
    8. **Unary Operators**
       1. $type, $lt $gt $or $and $multiply
    9. **Aggregation Pipeline**
       1. How does aggregation pipeline work?
       2. memory limit : 100mb
          1. spill to disk
    10. Batch sizing
    11. Iterator Size
    12. Query routing
    13. **Map Reduce**
        1. for what is it used?
        2. find sum, avg
34. **Indexes**
    1. pros and cons of Indexes
    2. createIndex({ *filed: value* })
    3. options when creating Index
       1. background: true
       2. unique: true
       3. name: “<*indexName*>”
    4. getIndex()
    5. dropIndex(), dropIndexes
    6. reIndex()
    7. rename Index
    8. hiding index
    9. covered query
    10. **Types of Indexes**
        1. Single Field Index
        2. Compound Index
        3. Multikey Index
        4. Text Index
        5. Geospatial, Hashed, Clustered Index
        6. Covered query
35. **Schema**
    1. pros and cons of using schema
    2. optional schema
    3. validation action
36. **Relationships**
    1. embedding
    2. referencing
    3. one-to-one
    4. one-to-many
    5. one-to-squillions
    6. many-to-many
37. **Replication**
    1. replica set
    2. advantage and disadvantages of replication
    3. **Replication Architecture**
       1. primary and secondary nodes
       2. arbiter
       3. process of election
       4. heartbeat
    4. Process of Election
    5. Replication lag
    6. operation log (oplog)
    7. **Types of replication**
       1. Asynchronous Replication
       2. Synchronous Replication
       3. Majority Commit
       4. etc…
38. **Sharding**
    1. advantages and disadvantages
    2. **Sharding Architecture**
       1. What is Mongos/Router
       2. Config Server
    3. **Types of sharding**
       1. Hashed sharding
       2. Ranged sharding
       3. Zone Sharding
    4. **Shard key**
       1. shard hotspots
       2. normal shard key
       3. hashed shard key
    5. Vertical and horizontal scaling
    6. Zones
    7. mongos
    8. auto balancer
    9. scatter-gather
39. **Cluster**
    1. types of cluster
    2. config servers
40. **Data Modeling**
    1. embedded data model
    2. reference data model
    3. linking vs embedding
41. **Transactions**
    1. How to do transaction
       1. **Session**
       2. startTransaction
       3. abortTransaction
       4. commitTransaction
    2. ACID Transaction
    3. A- Atomicity
    4. C- Consistency
    5. I - Isolation
    6. D - Durability
42. Create view in Mongodb
43. CAP Theorem
    1. theorem
    2. C- Consistency
    3. A - Availability
    4. P - Particle tolerance
44. **Isolation levels**
    1. Read Concerns
    2. - local
    3. - maojiry
    4. - available
    5. Write Concerns
    6. - w:1 (Acknowledged)
    7. - w:0 (Unacknowledged)
    8. - majority
    9. - all
    10. - journaled

## VS

* 1. $or vs $in
  2. $all vs $in
  3. $elemMatch vs $in
  4. drop() vs remove()
  5. findAndModify() vs findOneAndUpdate()
  6. Primary key vs secondary key
  7. join vs lookup
  8. dot notation vs nested form
  9. $currentDate vs $$NOW
  10. delete() vs remove()
  11. bulkWrite vs InsertMany
  12. replace vs update
  13. shard vs node vs cluster
  14. Aggregation Pipeline vs Map Reduce
  15. vertical scalability vs horizontal scalability
  16. load balancer vs sharding
  17. odm vs driver
  18. stage operator vs accumulator operator
  19. normal shard key vs hashed shard key
  20. aggregate([$count:”tota”]) vs find({}).count()
  21. replication vs replica set
  22. transaction vs query
  23. scaling up vs scaling down vs scaling out?
  24. config servers vs mongos
  25. load balancer vs auto balancer
  26. countdocument vs count

1. What is a MongoDB driver?
2. Capped collection and it’s advantages
3. Profiler
4. Explain
5. Soft deleting

## Interview Question

1. What to do when your quireing becomes slow?
2. What to do when your files are getting very big?
3. How to condense large volumes of data?
4. How to search for text in MongoDB?
5. How does MongoDB schema change?
6. How can we Backup and Restore in MongoDB?
7. What are the pros and cons of Normalising Data in MongoDB

## Good to Know

1. Atomicity
2. Type Bracketing
3. Dot Notation
4. Cursor behaviour
5. Aggregation Pipeline
6. Retryable Writes and Reads
7. MongoDB CRUD Concepts
8. B-Tree
9. ACID compliance
10. Mongoose
11. Network Components
    1. load balancer
    2. firewall
12. **CAP Theorem**
    1. consistency
    2. availability
    3. partition tolerance
13. Firewall
14. **Mongo Utilities**
    1. mongoexport
    2. mongoimport
    3. mongodump
    4. mongorestore
    5. mongostat
    6. mongotop
    7. mongooplog
15. Clustered collections
16. WAL

# React

## Set up

1. npx create-react-app <appName >
2. components
   1. default is App
3. rafce, tsrafce
4. calling function on button click
   1. without parameter
   2. with parameter
5. Fragments
6. Children Prop

## Theory

1. What is React
2. DOM
   1. DOM vs Virtual DOM
   2. Reconciliation
      1. working
   3. Diffing Algorithm
   4. React Fibre
      1. incremental rendering
   5. Shadow DOM
3. Dynamic rendering
4. props vs state
5. Server Side vs Client Side Rendering in React
6. Synthetic Events
   1. Event Pooling
7. Life Cycle
8. View Oriented
9. Memoization
10. Pure functions and components
11. Strict Mode
12. SPAs vs MPAs
13. CSR vs SSR
14. Static vs Dynamic rendering
    1. ISR, SPA
15. **Components**
    1. A React render tree
       1. top-level components
       2. leaf components
    2. Props
       1. immutable
    3. Forwarding props
    4. children
    5. Importance of making them pure
    6. local mutation
16. **JSX**
    1. Rules of JSX
    2. Fragment
    3. JavaScript in JSX
    4. HTML VS JSX
17. Conditional rendering
18. Key
19. **UI as a tree**
    1. Render trees
    2. Module Dependency Tree
    3. Bundler
       1. eg: Webpack
       2. Compiling
       3. Loader
       4. Code splitting
20. **Rendering steps**
    1. Triggering
    2. Rendering
    3. Committing
21. Rerendering
22. Batching updates
23. **State**
    1. Behaviour
    2. Queueing updates
    3. Updater function
    4. Updating object
    5. local var vs state var
    6. local mutation
    7. Lifting state
    8. Reducer
24. Declarative vs Imperative UI
25. **Event handlers**
    1. onClick, onSubmit etc…d
    2. Stopping propagation
    3. Preventing default
26. Lifecycle Methods
    1. What is Mounting, Unmounting
    2. **Phases**
    3. **- Mounting phase**
       1. constructor
       2. render
       3. getDerivedStateFromProps
       4. componentDidMount
    4. **- Updating phase**
       1. shouldComponentUpdate
       2. componentWillUpdate
       3. componentDidUpdate
          1. getSnapshotBeforeUpdate
    5. **- Unmounting phase**
       1. componentWillUnmount
    6. **- Error Handling**
       1. getDerivedStateFromError
       2. componentDidCatch
27. **Hooks**
    1. useState
       1. changeValue
       2. changeValueWithFunction
    2. useRef
       1. html
       2. useState vs useRef
       3. forwardRef
       4. useImperativeHandle
       5. flushSync
    3. **useEffect**
       1. dependency
       2. return in useEffect
       3. useLayoutEffect
    4. useMemo
       1. sample
       2. recache
       3. pros and cons
       4. referential equality
    5. useHistory
       1. push
       2. pop
       3. replace
       4. Redirect
    6. useNavigate
       1. navigate()
          1. route
          2. -1, 1
    7. useCallback
       1. sample
       2. useMemo vs useCallback
       3. uses
    8. useContext
       1. sample
    9. useReducer
    10. **Create custom hooks**
        1. useDebugValue
    11. useTransition
    12. useDeferredValue
    13. useId
        1. sample
    14. useImperativeHandle
28. **Props**
    1. default prop
    2. PropDrilling
    3. Children
29. **Components**
    1. Creating Components
    2. Controlled vs Uncontrolled Components
       1. Inputs
    3. Higher order components
    4. Pure components
30. **React Router**
    1. install
    2. **Hooks**
       1. useHistory
       2. useNavigate
    3. use
    4. **Link**
       1. replace
       2. reloadDocument
       3. state={}
       4. - useLocation()
       5. **NavLink**
          1. -isActive
          2. end
       6. **Navigate**
          1. useNavigate
          2. navigate(-1)
    5. **Types of Router**
       1. BrowserRouter
       2. HashRouter
       3. HistoryRouter
       4. MemoryRouter
       5. StaticRouter
       6. NativeRouter
    6. params (:id)
    7. cont {<name>} = useParams()
    8. useSearchParams
    9. **Nesting Routes**
       1. index
       2. location
       3. shared element with children
       4. outlet
       5. - useOutletContext()
       6. Nesting in separate file
       7. useRoute

## Good to Know

1. Immer
2. Object.entries(e)
3. Icons
4. Experimental Hooks
   1. useEffectEvent
   2. use
   3. useFormStatus
5. useOptimistic

## Week 2

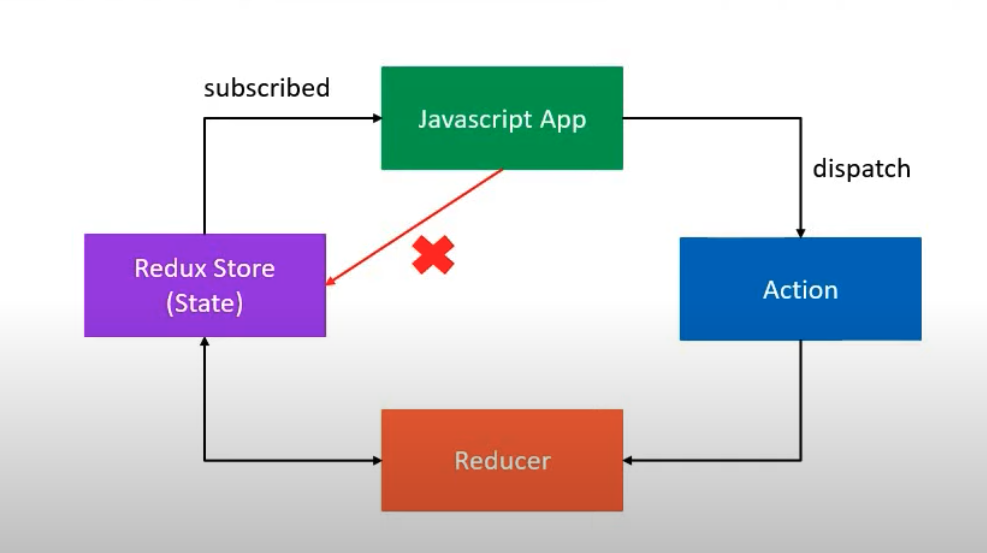
1. Render props
2. Higher order components
3. Custom hooks
4. Code splitting
   1. Route based
   2. Component based
   3. React.lazy
5. Higher order comps
6. **Lazy Loading**
   * 1. fallback ui
     2. suspense
     3. **Error boundaries**
     4. componentDidCatch
     5. Fallback UI
     6. Nested & Propagation
7. **useReducer**
   1. Dispatch
   2. useReducer vs useState
   3. useReducer vs redux
   4. payload
8. **PropTypes**
   1. types => name, string, any
   2. required, optional,
   3. node, element type
   4. oneof, shape
   5. PropTypes vs Typescript
9. **useMemo vs useCallback**
   1. React.Memo vs useMemo
   2. Object reference
   3. Pros and cons of memoization
10. **Context API**
    1. Provider
    2. Consumer
    3. useContext
    4. useReducer
11. **Webpack**
    1. Module Bundler
    2. Code Splitting
    3. Webpack Dev Server
    4. Hot Module Replacement (HMR)
    5. Tree Shaking
12. **Babel**
    1. Transpilation
    2. Plugins
    3. Runtime Polyfills
    4. Dynamic Import
13. useDeferedValue
14. dead code elimination
15. useTransition

## Others

* 1. forward ref
  2. useDebugValue
  3. useImperativeHandle
  4. Axios interceptor
  5. Concurrent Requests
     1. axios.all(), axios.spread()
     2. cancel Token

# Redux

## Theory

1. Why, what
2. Redux
3. How redux stores data
4. Architecture
5. Store
6. pros and cons
7. Redux store
8. Middleware
9. Calling APIs
10. React reducer vs Redux
11. **Store**
    1. Dispatch
    2. subscribe
       1. unsubscribe
    3. getState
    4. replaceReducer
    5. Store enhancer
12. **Action**
    1. Action creator
13. **Reducer**
    1. rules
14. **Redux flow**
15. **Redux principles**
    1. Store
    2. Action
    3. Reducer
16. Selectors
    1. Memoized selector
17. **Middleware**
    1. - Logger, crash reporting
    2. - Perform async tasks
    3. applyMiddleware
    4. Redux Thunk
       1. Thunk vs saga
       2. Payload creator
    5. Adding multiple middleware
18. **Slice**
    1. init state
    2. reducers
    3. extraReducers
19. **Redux toolkit**
    1. Nanoid
    2. Redux Query.
20. Normalising Data
    1. Normalised state
    2. createEntityAdapter
    3. shallowEqual, reference equality
21. Serializing
22. Hydrating
23. redux vs flux
24. saga vs thunk

## Other

1. Immer and the working of Immer in redux.
2. Access store outside of redux components
3. Flux by fb
4. Log rocket
5. createAsyncThunk
6. createEntityAdapter
7. createSelector
8. createListenerMiddleware

## JWT

1. What?
2. **Structure**
   1. Header
   2. Payload
      1. iat
      2. exp/eat
   3. Signature
3. Authentication working
4. Pros and cons
5. Expiration Time
6. Bearer token
7. Revocation
8. refresh token
9. Authentication vs Authorization
10. Types of Claims
    1. public
    2. registered
    3. private

# React Native

### Core Components

1. Text
2. View
   1. default flexbox layout
3. Button
   1. onPress
   2. onLongPress
   3. onPressIn
   4. onPressOut
4. Pressable
5. List
6. Flatlist
   1. optimises scroll performance
   2. Item separation
   3. key
   4. footer and header
7. Flatlist vs Map
8. Section list
9. ScrollView
10. SafeAreaView
11. Image
    1. ImageBackground
    2. react-native-svg
12. Modal
13. Alert
14. Switch
15. StatusBar
16. ActivityIndicator
17. TouchableOpacity
18. TouchableHighlight
19. TouchableWithoutFeedback

### Styling

1. style
2. StyleSheet Utility
3. NatvieWind
4. Inheritance
5. ios vs android
   1. border in text
   2. shadow
6. flexbox
   1. justify content
   2. align baseline
   3. align self
7. Relative and Absolute layout
8. **Dimension API**
9. Drawback of Dimensions API
10. useWindowDimensions
11. PlatformModule
    1. Platform.OS
    2. Plastform.Select
    3. using Extensions
12. **Navigations**
    1. Expo vs ReactNavigation
    2. Stack
    3. Drawer
    4. Tab

# DSA

### Algorithms

* + **Search**
  + Binary Search(recursive also)
  + Linear Search

1. Recursion
2. Iterative & recursive
3. Virtual memory
4. Amortised resizing
5. Dynamic programing
   * Memoize approach
   * Bottom up approach

### Problems

* + Factorial, fibonacci, prime number (with and without recursion)

### Complexity Analysis

* + Time complexity
  + Space complexity

### Asymptotic Notations

* + Ranking
  + Big O notation
  + Omega Notation
  + Theta Notation

### Memory

1. **Memory Allocation**
   * Bit vs byte
   * Memory address
   * Contiguous memory allocation
   * Non-contiguous memory allocation
   * **Stack** 
     1. Primitive types are stored in stack
   * **Heap**
     1. Reference type are stored in heap
     2. Eg: Arr, fun, obj
2. **Memory Leak**
   * Symptoms
   * **Garbage Collections**
     1. Process
   * Reasons for memory leak
   * How to debug
3. **Big O Notation**
   * Linear time complexity
   * Constant time complexity
   * Quadratic time complexity
   * Qubic
   * Logarithmic complexity
   * Exponential complexity
4. **Operations in normal array**
   * Init
   * Set
   * Get
   * Traverse
   * Insert
   * Delete

### Data Structures

1. What is DS?
2. Advantages and Disadvantages
3. Examples
   * DOM
   * Undu & Redo
   * Os job scheduling
4. **Dynamic Array**
   * It’s working and memory allocation?
   * Set
5. **Linked List**
   * Advantages and disadvantages
   * Applications
   * **Creating a linked list**
   * **Operation**
     1. Init
     2. Set
     3. Get
     4. Traverse
     5. Insert
     6. Delete
   * Singly Linked List
   * Double linked list
   * Circular linked list
   * Array vs linked list

## OTHERS

1. **Build in DS in JS**
   * **Array**
     1. Push, pop, shift, unshift, forEach, map, filter, reduce, concat, slice, splice ,sort()
     2. some(), every(), find(), findIndex(), fill(), flat(), reverse(), sort()
   * **Objects**
     1. Insert, Remove, Access, Search,
     2. Object.keys(), Object.values(), Object.entries()
   * **Sets**
     1. add, has, delete, size, clear
   * **Maps**
     1. set, get , has, delete, size, clear
   * Array vs Set
   * Object vs Map
   * **Strings**
     1. Primitive and object string
     2. Escape char
     3. ASCII
        1. 32 - Space
        2. 48-57 == (0-9)
        3. 65-90 == (A-Z)
        4. 97-122 == (a-z)
     4. Unicode
     5. UTF-8
2. **Custom DS**
   * Stacks
   * Queue
   * Circular queues
   * Linked lists
   * Hash tables
   * Trees
   * Graphs

## 

## Intermediate

### Algorithms

* + **Sorting**
  + Bubble sort
  + Insertion sort
  + Quick sort
    1. Divide and conquer
    2. Partition method
    3. **Pivot selection**
    4. Last, first
    5. average/median
  + Heap sort
  + Merge sort
    1. Divide and conquer
  + Merge vs Quick sort

### Data Structures

1. **Stacks**
   * LIFO
   * Push, pop
   * Stack underflow
   * Stack overflow
   * Use cases
   * **Types of Stack**
   * Linear Stack
   * Dynamic Stack
   * Array-based
   * Linked list based
   * Monotonic stack
2. **Queue**
   * FIFO
   * Enqueue
   * Dequeue
   * Peek
   * Priority queue
   * Circular queue
   * Uses
   * **Types of Queue**
   * - Linear Queue
   * - Circular Queue
   * - Priority Queue
   * - DEqueue (Double ended queue)
     1. Input restricted
     2. Output restricted
   * - Blocking Queue
   * - Concurrent Queue
   * - Delay Queue
3. **Hash Table**
   * Searching O(1)
   * Hash function
   * Collision
   * Dynamic restructuring
   * Uses
   * Load factor
   * **Operations**
   * Init
   * Insert
   * Search
   * Delete
   * Traverser
   * **Please Note**
   * Week set, week map
   * **Collisions Handling**
   * - Separate Chaining
   * - Open Addressing
     1. Linear Probing
     2. Quadratic Probing
     3. Double Hashing
     4. Clustering
   * - Cuckoo hashing
   * - Robin Hood hashing
4. **SHA: Secure Hashing Algorithm**

## 

## Advanced

1. Linear, non-linear, hierarchical

### Data Structures

1. **Tree**
   * Features
   * Uses
   * parent, child, root, leaf, sibling, ancestor, descendent, path, distance, degree, dept, height,edge,subtree
   * **Types of trees on nodes**
   * - Binary tree
   * - Ternary tree
   * - K-array tree
   * - Threaded binary tree
   * **Types of trees on structure**
   * - Complete tree
   * - Full tree
   * - Perfect tree
   * **- Degrenarted**
     1. Left-skew
     2. Right-skew
2. **Binary Search Tree (BST)**
   * BST vs BT
   * Uses
   * Balanced vs unbalanced tree
   * Properties of BST
   * **Operations**
   * - Inserting
   * - Deletion
   * **- Traversal**
     1. **DFS**
     2. - InOrder
     3. - PreOrder
     4. - PostOrder
     5. **BFS**
3. **Balanced Search Tree**
   * AVL tree
   * Red-black tree
   * Prefix tree
   * M-way search tree
   * - B Tree
   * - B+ Tree
   * Merkle Tree
   * Red-black tree vs AVL
4. **Heap**
   * Min Heap
     1. **To get value of**
     2. - Left child
     3. - Right child
     4. - Parent
     5. **Operations**
     6. - Init/ Heapify
     7. - Insert
     8. - Delete
   * Max Heap
   * Heapfity
     1. Bottom-up
     2. Top-down
   * DEPQ
5. **Trie**
   * String vs Trie
   * **Operations**
   * - Init
   * - Insertion
   * - Delete
   * - Search
   * Prefix and Suffix tree
   * - terminator char
   * **Compressed Trie**
   * - Radix Tree (Patricia Trie)
6. **Graph**
   * Vertex, Edge
   * Can be stored as
   * - Adjacency list
     1. as linked list
     2. time O(V)
     3. space O(V+E)
   * - Adjacency matrix
     1. As array
     2. time O(1)
     3. space O(v^2)
   * Spanning tree
     1. min spanning tree
   * Graph indexing
     1. Vertex-centric indexes
     2. Edge-centric indexes
   * **Types**
   * - Unidirectional (Direct graph)
   * - Bidirectional (Un DIrected graph)
   * - Cyclic
   * - Disconnected
   * - Weighted Graph
   * - Unweighted Graph
   * - Bipartite Graph
   * **Traversal**
     1. BFS
     2. DFS
   * River size problem

### Algorithms

1. Greedy method
2. Kruskal's Algorithm
3. Prim's Algorithm
4. Dijkstra's Algorithm
5. Bellman-Ford Algorithm
6. Topological Sorting
7. Floyd-Warshall Algorithm
8. Bipartite Graph Checking
9. Max Flow (Ford-Fulkerson Algorithm)

### Question

1. Graph vs Tree
2. Forest (in Tree)
3. Forest > Graph > Tree > Linked list
4. Operators
   * Binary operators
   * Priority
   * Infix
   * Prefix (Polish notation)
   * Postfix (Reverse Polish notation)

### General

1. How does Logarithms work
2. File structure vs Data Structure
3. Where is the DS used?
4. Void vs null
5. Dynamic data structure
   1. Uses
   2. Example
6. Dynamic memory management/ allocations
7. Heap be used over a stack
8. Data abstraction
9. Post fix expression
10. Signed number
11. Pointers in DS
    1. Uses
12. Huffman’s algorithm working
13. What is recursive algorithm
    1. Divide and conquer on recursion
14. Which is the fastest sorting algorithm available?
15. Multi linked
16. Sparse matrices
17. Disadvantages of implementing queues using arrays
18. Void pointer
19. Lexical analysis
    1. Lexeme
    2. Pattern

# Hosting

### Nginx

1. **Commands**
   1. systemctl nginx status
   2. restart and reload
2. Contex
   1. Eg: http, events, server
   2. Worker process and connection
   3. Directive & block
   4. Location block
      1. root, alias, try\_files
3. Master Process
4. Worker Process
5. Firewall
6. DDOS protection
7. K8s IC
8. Sidecar proxy
9. Virtual host
10. Brute force
11. WAF
12. UFW
13. TCP vs UDP
14. TCP vs TCL
15. **Load Balancing**
    1. Round robin
    2. Least connection
    3. IP hash
16. Caching
17. **Proxy**
    1. Proxy server
    2. Reverse proxy
    3. Forward proxy
    4. Load balancer vs reverse proxy
18. Nginx vs Apache
19. Working of HTTPS

### SSH

1. How does it work??
2. Private key
3. Public key

### SSL

1. How does it work??

### Linux

1. apt
2. rm
3. mkdir
4. touch
5. mv
6. nano
7. more, less
8. head, tail
9. >, <
10. /
    1. bin
    2. boot
    3. dev
    4. etc
    5. home
    6. root
    7. lib
    8. var

# System Design

1. Dark scale distributed system
2. Scaling
   1. Vertical
   2. Horizontal
   3. Auto
      1. pros and cons
3. Asynchronous system
   1. Queue system
4. Scaling database and server
5. Rate limiter vs Limiter
6. **API Rate Limiting**
   1. Token Bucket
      1. pros and cons
   2. Leaky Bucket
7. Concereny controller
8. Handling Failure

### Theory

1. **Components of System Design**
   1. Logical
      1. Data
      2. Database
      3. Users
      4. Applications
      5. Cache
      6. Communication protocol
      7. Infra
      8. Message queues
      9. Presentation layer
   2. Tangible
      1. data - Text, image
      2. database - SQL, NoSQL
      3. App - Java, node
      4. Cache - Reddist, mem-cache
      5. Infra - AWS, GCP
      6. Comm - API, RPC, Message
      7. Queues - Kafka, RabbitMQ
2. **Client Server Arch**
   1. Thick and Thin client
   2. 2-Tier, 3-Tier, N-Tier Client
3. **Fault and failure**
   1. Fail safe
   2. Fault tolerant
   3. graceful fail
   4. Testings
   5. Transient vs Permanent fault

### Other

1. Critical and non critical tasks

# Git

### THEORY

1. **Centralised** Version control system vs **Distributed** Version control system
2. Config
3. Working directory
4. Staging area
5. git init
6. git clone
7. git status
8. git log
9. **Creating Version**
   * git add *file*
     1. git add - - all
     2. git add .
   * **git commit**
     1. -m “<*message>*”
     2. Commit without staging
   * commit id
     1. check sum
     2. **content**
        1. author details
        2. preview details
        3. date
        4. etc..
     3. sha-1 hash
   * label
   * **branch**
10. touch
11. **git log**
    * git log
    * git log - - all
    * git log –p -1
    * git log graph
12. git diff
13. git diff –staged
14. **Restore**
    * git restore
    * git restore –staged
15. **Branching**
    * git branch <*branchName>*
    * git branch
    * git branch —all
    * Creating branch
    * Deleting branch
    * git checkout vs git switch
    * switching b/w branches
    * commit id
    * branch name
16. **Stashing**
    * git stash st
    * git stash apply
    * git stash drop
    * git stash list
17. **Merging**
18. git merge <*branchName>*
19. **Types of merging**
    * fast-forward merge
    * **recursive merge**
      1. conflict
20. **Git server**
    * git remote add <*name> <url>*
      1. git remote
      2. git remote -v
    * git push <*remoteName> <branchName>*
    * git push set upstream
    * **Cloning**
    * git clone <*url>*
    * git pull
    * pull vs pull request?
    * pull vs fetch
21. **Tags**
    * Simplified
    * Annotated
    * git tag
    * Should Pushing tags
22. **Forking**
23. git rebase
24. vim .gitignore
25. gist
26. **ci cd**
27. git projects

### GOOD TO KNOW

1. rebase
2. tree

# SQL: Postgres

### Theory

1. SQL vs NoSQL (Relational vs non-relational)
2. Web-scaled
3. When to use SQL and NoSQL
4. Expression, Statement, Operators
5. **Data types SQL**
   1. null, bit
   2. int, real / float
   3. char, varchar, text
   4. boolean
   5. date, datetime, timestamp
   6. xml/json
   7. – char vs varchar vs text
   8. – datetime vs timestamp
   9. – JSON vs JSONB
6. **Operators**
   1. Arithmetic, Logical, Comparison, Bitwise
7. Primitives: Integer, Numeric, String, Boolean
8. Structured: Date/Time, Array, Range / Multirange, UUID
9. Document: JSON/JSONB, XML, Key-value (Hstore)
10. Geometry: Point, Line, Circle, Polygon
11. Customizations: Composite, Custom Types

### Postgres

1. **complex queries**
   1. Aggregation
   2. Subquery
   3. Window Function
2. **foreign keys**
3. **triggers**
   1. Trigger Timing
   2. - BEFORE and AFTER
   3. - Uses
4. **updatable views**
5. **transaction integrity**
6. **multiversion concurrency control**
7. **functions**
   1. Stored Procedures
   2. Window functions
   3. Aggregate functions
8. **operators**
9. **aggregate functions**
10. **index methods**
11. **procedural language**
12. Forks
13. client/server model
14. **Data types Unique to Postgres**
    1. interval
    2. point
    3. bigserial
    4. etc…
15. Database cluster
16. **Constraints**
    1. UNIQUE
    2. NOT NULL
    3. PRIMARY KEY
       1. as UUID
    4. FOREIGN KEY
    5. CHECK (<condition>)
    6. - Adding & removing constraints after creating table

### Commands

* 1. list db
  2. to connect
  3. list tables
  4. Move to super
  5. list specific table
  6. List current table

1. Creating
   1. Database
   2. Table
2. Drop
   1. Drop DB
   2. Drop Table
   3. Drop constraints
3. Commands
   * 1. – or /\* \*/
   1. **Database migration**
      1. Add, Delete, Migration
      2. Up migration
      3. Dow migration

### Functions

* 1. SELECT
     1. LIMIT
     2. FETCH
     3. OFFSET
     4. AS
     5. DISTINCT
     6. GROUP BY
        1. HAVING
        2. GROUPING SETS
        3. ROLLUP
        4. CUBE
     7. Having vs Where
     8. Limit vs Fetch
  2. FROM
  3. WHERE
     1. AND, OR
     2. LIKE, ILIKE
     3. BETWEEN
     4. IN
     5. IS NULL, IS NOT NULL
  4. ORDER BY
     1. DESC, ASC
  5. DELETE
  6. DELETING FOREIGN KEY
     1. CASCADE
  7. UPDATE
     1. SET
  8. RENAME COLUMN
  9. **JOIN**
     1. INNER JOIN
        1. ON
     2. LEFT JOIN
     3. RIGHT JOIN
     4. FULL JOIN (FULL OUTER JOIN)
     5. SELF JOIN
     6. CROSS JOIN
     7. NATURAL JOIN
  10. **VIEWS**
      1. Pros and Cons
      2. CREATE VIEW
      3. Materialized View
         1. Write amplification
  11. UNION
  12. COALESCE
  13. NULLIF
  14. Index
      1. multi index

1. AUTO\_INCREMENT
2. ON CONFLICT
   1. DO NOTHING
   2. **Upserting**
   3. - DO UPDATE
      1. EXCLUDED
3. **Date functions**
   1. INTERVAL vs AGE
4. **Aggregate functions**
   1. AVG, MIN, MAX, SUM, ROUND, COUNT, CONCAT
5. **Scalar Functions**
   1. LCASE, CASE, LEN, MID, ROUND, NOW, FORMAT ,
   2. INITCAP , LEFT , RIGHT , CONCAT , ABS , CEIL , FLOOR,
   3. UPPER AND LOWER in psql.
6. Aggregate vs Scalar
7. **Window function**
   1. OVER
   2. - PARTITION BY, RANK, LEAD, LAG
   3. CASE
8. **SQL Commands**
   1. **DDL**
      1. CREATE, ALTER, DROP, TRUNCATE
      2. DROP vs TRUNCATE
   2. **DML**
      1. INSERT, SELECT, UPDATE, DELETE
   3. **DCL**  
       GRANT, REVOKE
   4. **TCL**
      1. COMMIT
      2. ROLLBACK
      3. SAVE POINT
   5. **DQL**
      1. SELECT
9. **3-Schema architecture**
   1. Internal level
   2. Conceptual level
   3. External level
10. BIGINT VS BIGSERIAL
11. **Combining queries**
    1. UNION, UNION ALL
    2. INTERSECT, INTERSECT ALL
    3. EXCEPT, EXCEPT ALL
12. **Normalisation**
    1. **Levels**
       1. 1NF, 2NF, 3NF etc..
       2. BCNF
    2. **Anomalies**
    3. - Insertion anomalies
       1. Data redundancy
       2. Missing data
    4. - Deletion anomalies
       1. Losing data
    5. - Updation anomalies
       1. inconsistency
       2. Updating values on so many records unnecessarily
13. **Relationship**
    1. one to one
    2. one to many
    3. many to may
14. **Transaction & ACID**
15. **- Transaction**
    1. COMMIT
    2. ROLLBACK
    3. SAVE POINT
       1. RELEASE SAVEPOINT
    4. LOCK
       1. Exclusive Locks (X-Locks)
       2. Shared Locks (S-Locks)
       3. Update Locks (U-Locks)
       4. Intent Locks
       5. Read and Write Locks
16. **- ACID**
    1. - Atomicity
    2. - Consistency
       1. Consistency in data
       2. Consistency in reads
    3. - Isolation
       1. **Read phenomena**
       2. - Dirty reads
       3. - Non-repeatable reads
       4. - Phantom reads
          1. Serialotions
       5. - (Lost updates)
       6. **Isolation level**
       7. - Read uncommitted
       8. - Read committed
       9. - Repeatable Reads
       10. - Transactions are Serialized
    4. - Durability
    5. How to implement ACID properties
17. EXPLAIN
18. Heap Scan
19. Parallel Scan
20. Planner

### Other theory and functions

1. COPY
2. OLTP
3. MUCC

### Pendings

1. Delete vs truncate
2. candidate key vs super key
3. stored procedure
4. ER diagram.
5. Practice nested queries.

# Microservice

## Concepts & Theory

1. What is a service?
2. Monolithic arch
   1. pros and cons
3. Microservice arch
   1. pros and cons
4. **Monolithic vs Microservice**
   1. deployment, scalability, reliability, development, flexibility, debugging
5. Security
6. **Cloud computing**
   1. Public IP address
   2. On-premises
   3. Iaas, Cass, Pass, Faas (Server less computer), Saas
   4. Private could
   5. Hybridge cloud
7. Scaling
8. Blue Green Deployment
9. Cloud Native vs Cloud Ready
10. Event-Driven Architecture
    1. Event producer
    2. Event broker
    3. consumer
    4. pub/sub
    5. eventual consistency
    6. cache layer
    7. idempotent
11. 12 Factor App
    1. Codebase
    2. Dependencies
    3. Config
    4. Backing services
    5. Build, release, run
    6. Processes
    7. Port binding
    8. Concurrency
    9. Disposability
    10. Dev/prod parity
    11. Logs
    12. Admin processes
12. Load balancing
    1. Round robin
    2. Least connection
    3. IP hash
13. Service Registry
14. Failed fast
15. Service Discovery
16. **Tools**
    1. os
    2. language
    3. api management
       1. postman
    4. messaging
       1. kafka
       2. rabbitMQ
    5. toolkits
       1. fabric8
       2. seneca
    6. orchestration
       1. kubernetes
       2. Istio
    7. monitoring
       1. prometheus
       2. logstash
    8. serverless tools
       1. claudia
       2. AWS lambda
17. **Principles behind microservices**
    1. Independent and autonomous service
    2. Scalability
    3. Decentralisation
    4. Resilient services
    5. Real time load balancing
    6. Availability
    7. CICD
    8. Continuous monitoring
    9. Seamless API integration
    10. Isolation from failures
    11. Auto provisioning
18. **Security**
    1. Defence in depth mechanism
    2. Token and API gateway
    3. Distributed tracing
    4. First session
    5. Mutual SSL
    6. OAuth
19. API gateway
    1. client performance
    2. security
    3. rate limiting
    4. monitoring logging
    5. BFF
20. SOA vs Microservices
21. **Communication**
    1. Types
       1. synchronous blocking communication
       2. asynchronous non blocking communication
    2. Request response
       1. REST over HTTP
       2. RPC
    3. Event driven
       1. kafka

## Design Patterns

1. need?
2. Aggregator
3. **API gateway**
4. Chained or chain of responsibility
5. Asynchronous messaging
6. Orchestration vs Choreography
7. **Database pattern**
   1. Database Per Service
   2. Shared Database
8. Event sourcing
9. Branch
10. Multi-tenant
    1. pros and cons
11. **CQRS**
12. **Circuit breaker**
13. SAGA
    1. Choreography
    2. Orchestration
14. Decomposition
    1. Vine or Strangle
15. **Database**
    1. Decentralised Data Management
       1. pros and cons
    2. **Data Consistency in microservice**
       1. Saga Pattern
       2. Event-Driven Architecture
       3. CQRS
       4. Idempotent Operations
       5. Consistency Models
    3. Database per Microservice
    4. Shared Database
    5. Data Virtualization
    6. Distributed Data Mesh
16. **CI/CD**
    1. Github actions
    2. pros and cons
    3. running in parallel
    4. **Testing**
       1. unit tests, integration tests, and end-to-end tests.
    5. Artefact Repository
       1. JFrog
17. **Github actions** 
    1. Workflows
    2. Events
    3. Jobs
    4. Actions
    5. Runners
    6. Using variables in your workflows
    7. Sharing data between jobs
       1. artefacts
          1. actions/download-artifact
    8. Literals
    9. Contexts
       1. uses
       2. Context availability
       3. github context
       4. env context
       5. var context
       6. job context
    10. Polyglot Persistence
18. **- commands**
    1. name
    2. on
       1. push
          1. branches
    3. jobs
       1. needs
       2. steps
       3. uses
       4. with
       5. run
       6. if
       7. matrix
       8. outputs
19. **Transactions in microservice**
    1. Two-phase commit
       1. voting phase
       2. commit phase
       3. pros and cons
    2. SAGA
       1. backward recovery
       2. forward recovery
    3. correlation id
    4. imp of logging and monitoring

# 

# 

# Docker

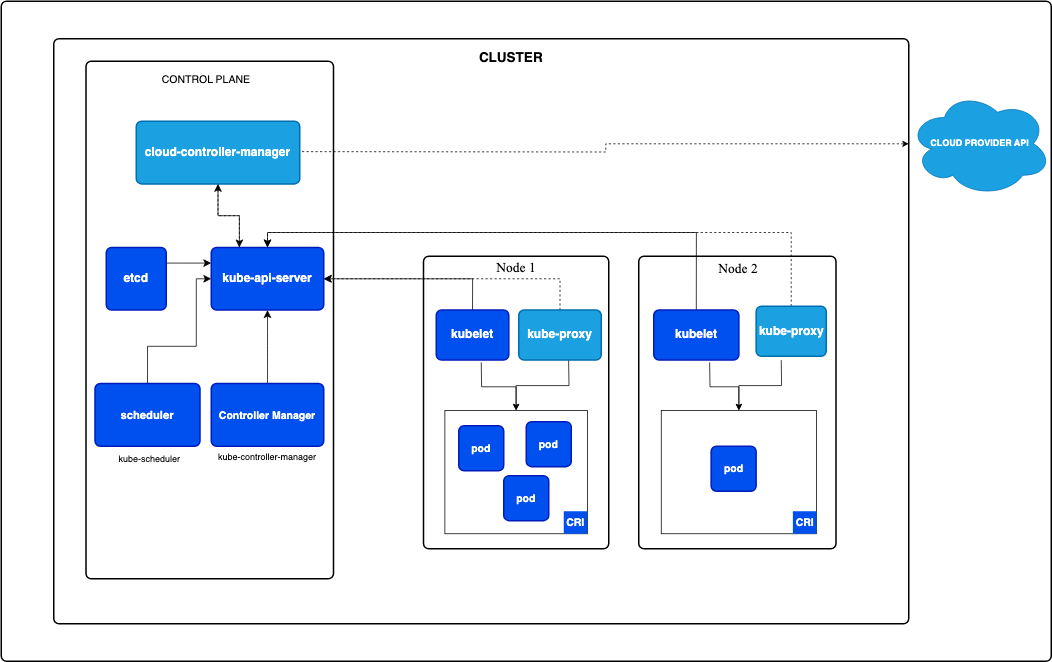
1. What, Why, When
2. Architecture
   1. client and server
   2. - server => docker engine
3. Container
   1. kernel namespaces
   2. C groups
   3. Container vs Virtual machine
4. Images & Container
   1. image vs container
   2. Isolated process
5. **Images**
   1. Image layers
   2. - base image layer
   3. - instruction layers
   4. - writable container layer
   5. Layer caching
6. docker run <ubuntu> vs docker pull <ubuntu>
7. Port mapping
8. Data persistence
9. DB Migration
10. Bind mounts.
11. run, start, rm
12. -t, -p

## Commands

1. docker init
2. docker tag
3. docker build
   1. -t
   2. buildx
4. docker run
   1. --name
   2. -it
   3. -e
   4. -d
   5. -p
      1. port mapping
   6. --net
   7. --rm
5. docker container
   1. ls
   2. stop
      1. -t
   3. prune
   4. rm
      1. -f
6. docker logs <container>
   1. --follow/ -f
7. docker image
   1. ls
   2. history
      1. --no-trunc
8. docker network
   1. ls
   2. create <name>
      1. -d
      2. --subnet
      3. --gateway
9. **Manage containers**
   1. Docker container ls || docker ps
   2. Docker container ls -a || docker ps -a
   3. \* Start
   4. \* Stop
   5. \* Restart
   6. \* rm
   7. Docker system prune -a
10. **Network commands**
    1. Docker network ls
    2. Docker inspect bridge
11. **Volume**
    1. types
    2. - bind mounts.
    3. - volume mounts/ named volumes
    4. bind vs named mounts
    5. scratch space
    6. Volume claim
    7. docker volume
       1. create
       2. inspect
    8. docker rm -f
12. dockerignore
13. **Docker hub**
    1. docker
       1. pull
       2. push
       3. rmi
14. **Docker compose**
    1. docker compose
       1. up
       2. down
       3. watch
       4. ps
    2. services
       1. image
       2. ports
       3. environment
       4. restart
          1. always
          2. on-failure
          3. unless-stopped
       5. depends\_on
       6. resources
          1. limits
          2. reservations
       7. volume mapping
          1. read only, write only
    3. networks
    4. secrets
    5. volumes
       1. driver
15. **Dockerfile**
    1. FROM
    2. COPY
    3. WORKDIR
    4. RUN
    5. CMD
    6. EXPOSE
    7. ENTRYPOINT
    8. ENV
    9. ARG
    10. USER
    11. LABEL
    12. RUN VS CMD
16. **Docker network**
    1. Bridge
    2. Host
    3. None
    4. overlay
    5. macvlan
    6. IPvlan
17. Docker daemon

# 

# Kubernetes



1. aka k8s
2. pros
   1. other pros from doc
3. imperative vs declarative
4. self heading/ auto-heal
5. scaling, auto-scale
   1. HorizontalPodAutoscaler
6. cluster
7. context
8. namespaces
9. annotation
10. namespaces vs annotation vs labels
11. Finalizers
12. Node
    1. master node
    2. worker node
    3. node pool
    4. Node status
    5. Node heartbeats
    6. Node controller
       1. what it does
       2. CIDR block
    7. Node topology
    8. Graceful node shutdown
       1. grace period
       2. non-graceful shutdown
13. **Pod**
    1. communicate via
    2. ephemeral
    3. atomic
    4. scaling
    5. **Pods life cycle**
       1. when creating
       2. when deleting
          1. grace period
    6. **Pod state**
       1. pending
       2. running
       3. succeeded
       4. failed
       5. unknow
       6. CrashLoopBackOff
    7. init container
    8. **Multi container pods**
       1. sidecar pattern
       2. ambassador pattern
       3. adaptor pattern
14. Container
    1. Images
    2. - Serial and parallel image pulls
    3. - image pull policy
    4. Container Environment
    5. Container Lifecycle Hooks
       1. PostStart
       2. PreStop
15. Kubelet
16. Selectors
    1. metadata > labels
    2. spec > selector
17. **Workloads**
    1. pod
    2. replicaSet
       1. self-heading
       2. template
    3. deployment
       1. replicas
       2. revisionHistoryLimit
       3. **Strategy**
          1. **RollingUpdate**
          2. - maxSurge
          3. - maxUnavailable
          4. - default
          5. - rollback
          6. - rollout
          7. **Recreate**
    4. daemonSet
       1. daemon controller
       2. uses
       3. spec > toleration
    5. statefulSet
       1. persistent identifier
       2. creation & deletion
       3. uses
       4. headless service
    6. job, cron job
    7. replicaSet vs deployment
    8. pods vs deployment
18. Volumes
    1. persistent volume
       1. claim
       2. HostPath
       3. drawback
       4. reclaim policies
          1. delete (default)
          2. retain
       5. access modes
          1. ReadWriteMany
          2. ReadOnlyMany
          3. ReadWriteOnce
       6. states
          1. available
          2. bound
          3. released
          4. failed
    2. storage class
    3. static and dynamic
19. Objects
20. ConfigMap
    1. static
    2. solve static with volume
21. Secret
    1. type
22. **Service**
    1. clusterIP
       1. port
       2. targetPort
    2. nodePort
    3. load balancer
       1. L4
       2. round robin
    4. ingress
       1. L7
23. NodePort
24. **k8s Cluster arch** 
    1. **Node**
       1. container runtime
          1. containerized
          2. CRI-O
       2. kubelet
       3. kube proxy
    2. **Control Plane / Master node**
       1. kube-api server
       2. kube-scheduler
          1. factor when scheduling
       3. Kube controller manager
          1. built-in controllers
          2. Node controller
          3. job controller
          4. endpointSlice controller
          5. serviceAccount controller
       4. Cloud controller manager
       5. ETCD
       6. **Addons**
       7. - DNS
       8. - WEBUI (dashboard)
       9. - cluster level logging
       10. - container resource monitoring
25. Cluster > Node > pod > container
26. CRI
27. Garbage Collection
28. Mixed Version Proxy
29. KubeCTL
30. Minikube
    1. rollout
31. Open Service Broker.
32. Ingress
33. Docker Swarm vs Kubernetes

### Security

1. **Image**
   1. Untrusted registries
   2. Vulnerabilities in tools of OS or libraries
2. Authentication & Authorization
3. practices
   1. use linear images
   2. image scanning
   3. don’t use root user
   4. manage user and permission
      1. RBAC
4. statefulSet
   1. master
   2. slave

### Yaml

1. apiVersion
2. kind
3. metdat
   1. name
   2. label
   3. namespace
4. spec
   1. containers

### Commands k8s

* 1. alias k=kubernetes
  2. k get
     1. pods
     2. svc
     3. deploy
  3. k delete -f <deployment.yaml> -f <service.yaml>
  4. k exec <pod> – nslookup <svc>

1. k config
   1. current-context
   2. get-contexts
   3. use-context <name>
   4. delete-context <name>
2. namespace
   1. k get ns or namespace
   2. k create ns <name>
   3. k delete ns <name>
   4. k config set-context --current --ns=<namespace>
   5. k get pods -n <namespace>
3. node
   1. k get nodes
   2. k describe node
4. Probes
   1. startup
   2. readiness
   3. liveness

## Good to know

1. grep
2. docker compose watch - <https://www.youtube.com/live/I-htDVxmFGM?si=5Um3NCnMi0BeAgCz>
3. chroot
4. Service Mesh

# 

# Message Broker

## Kafka

1. used as key value but stored as binary in kafka
2. default port
3. serialisation and deserialization
4. pros and cons
5. Kafka cluster
   1. Fault Tolerance
   2. Scalability
   3. Distributed Processing
6. **Kafka Broker**
   1. topics
      1. compacted topics
   2. partitions
      1. leader
      2. follower
      3. replication
         1. replication factor
         2. key
   3. segments
7. **Producer**
   1. record
      1. header
      2. key
      3. value
      4. timestamp
   2. retention period
   3. ack /nack
      1. no acks
      2. leader acks
      3. all acks
8. **Consumer**
   1. Queue vs Pub Sub
   2. Consumer group
9. Offset
10. Connectors
11. At most once
12. At least once
13. Exactly once
14. Exactly-Once Semantics
    1. Idempotent
    2. Two-Phase Commit
    3. alt
15. Persistent storage
16. Steam processing
17. Distributed system
    1. leader
    2. follower
    3. zoo keeper
       1. Metadata Management
       2. Leader Election
       3. Synchronisation
       4. Heartbeats and Timeouts
       5. Monitoring
       6. default port
       7. gossip
18. long polling
19. Kafka Connect

## 

## RabbitMQ

1. TCP
2. HTTPv2
3. AMQP
4. RabbitMQ server
   1. default port
   2. Exchange Queues
5. Heartbeats
6. Connection pool
7. Channels
   1. Multiplexing
   2. Concurrency
8. Message TTL
9. Message Acknowledgment
   1. **Strategies**
   2. Automatic Acknowledgment (Ack)
   3. Positive Acknowledgment
   4. Negative Acknowledgment (Nack)
   5. Rejection with Requeue
   6. Rejection without Requeue
10. **Exchanges**
    1. Fanout exchange
       1. pros and cons
       2. uses
    2. Direct exchange
       1. pros and cons
       2. uses
    3. Header exchange
       1. pros and cons
       2. uses
    4. Topics exchange
       1. pros and cons
       2. uses
    5. Dead Letter Exchanges and Queues
11. Polyglot persistence
12. Durability
    1. Durable Queues
    2. Persistence message
    3. Combined Durability
    4. rabbitMQ
13. Routing Key
14. Request response
    1. architecture
    2. breaks
    3. pros and cons
15. Publish subscribe (pub/sub) model
    1. Queue/Channels/Topics
    2. Publisher/producer
    3. Consumer
    4. pros and cons
16. Multiplexing
17. Channel
18. Push model

## gRPC

1. why?
2. http
3. protobuffer
4. Unary gRPC
5. Server streaming
6. Client streaming
7. Bidirectional

# TypeScript

### Git Repo

[Fore more info click here](https://github.com/raazi-muhammed/learning-resourses/tree/main)

### Theory

1. What is typescript
2. Disadvantages
3. Statically typed language
4. **Compiling project**
   1. tcs index.ts
5. setting type
   1. let age: number 20
6. Types
   1. implicit types an explicit types
   2. any type
   3. You will lose type case (It's not recommend to use any)
   4. unknown
   5. never
   6. enum
   7. Tuple
7. Objects
   1. Readyone
   2. Method
   3. Specitif valus
   4. Return type
8. Type alias
9. Union type
10. Type intersection
11. Literal types
12. Nullalbe type
13. Optione property, element, call
14. Interface
    1. Reopening interface
    2. Inheritance
15. Class
    1. Modifiers
    2. Getters and setters
    3. Abstand class
    4. Overrifdienr
    5. Diff b/w class and abstand class
16. Generics

# Next.js

### Theory

1. Prerendering
   1. SSG (Static site generation)
   2. SSR (Server side rendering)
   3. Suspense SSR Arch
      1. HTML streaming
      2. Selective hydration
   4. ISR (Incremental site generation)
   5. RSC (React server components)
   6. Pros and cons
2. **Routing**
   1. file based
   2. app based
   3. how to route
   4. dynamic route
   5. Catch all segments [...<slug>]
      1. optional catch all [[...]]
   6. Navigation
      1. Link component
         1. replace
      2. usePathname
         1. startWith
      3. useRouter
         1. push()
         2. replace()
         3. back()
         4. forward()
   7. Parallel Routes
      1. slots (@)
      2. pros and cons
      3. default.tsx
   8. Conditional Routes
   9. Intercepting Routes
      1. (.)<route>
      2. (..)<route>
      3. (..)(..)<route>
      4. (...)<route>
3. **Routing metadata**
   1. why?
   2. static vs dynamic metadata
   3. priority
   4. layout vs page metadata
   5. title metadata
      1. absolute
      2. default
      3. template
4. **Pages**
   1. not-found.tsx & notFound()
   2. loading.tsx
   3. error.tsx
      1. Error boundary
      2. error object
      3. reset
      4. error bubbling
   4. File colocation
   5. private folder
      1. \_
      2. advantages
      3. %5F
   6. Route groups
5. **Layout**
   1. nested layout
   2. route group layout
6. **Templates**
   1. why?
   2. templates vs layout
   3. using both
7. **Component hierarchy**
   1. Layout > Template > Error Boundary > Suspense > Erro Boudy (not found) > Page
8. Route Handlers
9. RSC (React server component)
10. API routes
11. Rending
    1. client side
    2. server side
12. Date fetching
13. STyling
14. Optimization
15. Layouting
16. Loading state
17. Error bordering
18. SEO
    1. Metadata
19. Fetching data
    1. Using server comp
    2. In parallel
    3. Fetch data where It’s used
    4. Streaming and suspense
20. Deduplication
21. Caching
    1. ISR (Incremental site generation)
    2. {cache: force-cache}
    3. {cache: no-store}
    4. {next: {revalidate: 60}}
22. Dynamic params

# NestJS

1. What
2. Why use NestJS
3. Module
   1. Global modules
   2. Third party modules
4. Dependency injections
5. **Decorators**
   1. Decorator factor
   2. TS vs JS
   3. Call, apply, bind
   4. This context in different functions
6. Bootstrap
7. Nest CLI
   1. create module
8. Controllers
9. Providers
10. Injectables
11. Nest validation
12. Pipes
    1. types of pipes
13. Class validators
    1. pipes vs class validators
14. Accessing code expires req
15. Strategy
16. Guards
    1. global
    2. local

**Blockchain**

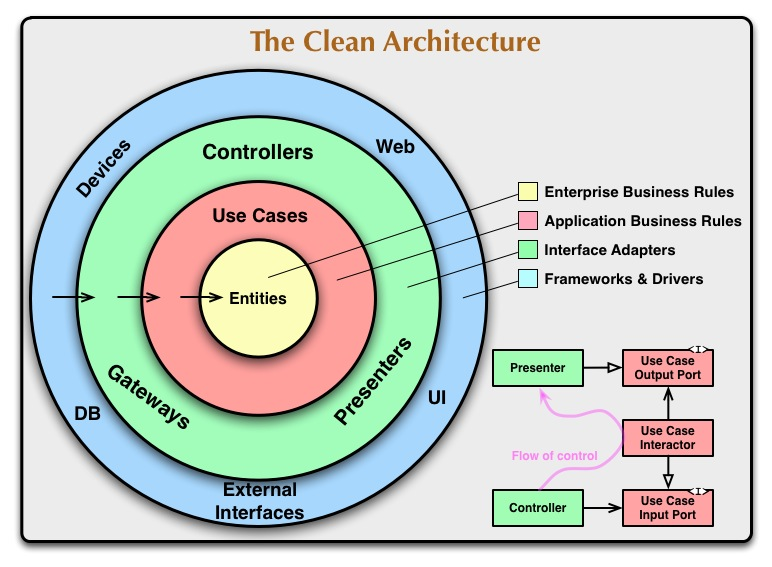
1. Centralised vs Decentralised
2. Blockchain
3. Ledger
4. Block
   1. data,
   2. hash,
   3. prev hash,
   4. Genesis block
5. Security
   1. Proof of work
      1. proof fo stake
   2. Consensus Rule
      1. Copy of chain in network
   3. Cryptographic proof
   4. Algo
      1. SHA256
      2. RSA
      3. MD5
6. Crypto
   1. no conversion
   2. centralised
   3. use case
      1. patient records
      2. real estate - lottery
7. Smart contracts
8. Ethereum
   1. Solidity
9. Mining
10. Gas fees

# Clean Code

1. You are not done when it work
2. Invest the time to spend to write the program to make the program clean
3. Clean code what is expect when to read the code
4. Function should be verb (not noun)
5. **Function**
   1. Every things in the function should have the same abstraction
   2. Functions should be small
   3. Function should not have more than 3 params
   4. Don’t pass boolean to a function
   5. Avoid switch statement
   6. The should not any side effect
   7. If a function return void, it should have side effects
   8. if a function returns a value, it should not have side effects
6. File should be <100 lines

### SOLID Design Principles

1. - Single responsibility
2. - Open-closed
3. - Liskov substitution
4. - Interface segregation
5. - Dependency inversion

****

**Clean Architecture**

### Things

1. Dependency Inversion Principle
2. Interface adapters
3. Entities
   1. They have no dependency
4. Use cases
   1. they only depend on entities
   2. Interactor
   3. Interface
5. Controllers
6. Gateway
7. Presenter
8. Devices
9. Web
10. Database
11. UI
12. External Interface

### Related Topics

1. Dependency Injection

### Rules

1. Data flow from outside to inside

### Videos

1. [Using Clean Architecture for Microservice APIs in Node.js with MongoDB and Express](https://youtu.be/CnailTcJV_U?si=dEdpC0xiNNt2cPoS)

# Python

### Basic

* Syntax
* Variables and Data Types

1. Integers
2. Floats
3. Strings
4. Booleans
5. Lists
6. Tuples
7. Dictionaries
8. Sets

* Type casting
* Slicing
* Scope of variables
* Operators

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Assignment Operators
5. Bitwise Operators
6. **Control Flow**

* Conditional Statements
  + If
  + elif
  + else
* Loops
  + for loop
  + while loop
  + break, continue, pass
* Comprehensions
  + List Comprehensions
  + Dictionary Compre…
  + Set Compre…
* Exception Handling
  + Try
  + except
  + finally
  + Else
  + raise
* Exception Handling

1. **Function**

* Function
  + docstring
  + return
* Lambda Functions
* Types of function arguments
  + Defualt argument
  + Keyword argument
  + Positional Arguments
  + Arbitrary Keyword Arguments
* call by sharing
* Genarator
* Decorators
* Recursion
* Map
* Filter
* Reduce
* eval

1. **Modules and Packages**

* Importing Modules
* Creating Modules
* Using Packages
* init.py
* Standard Library Modules

1. **File Handling**

* Reading Files
* Writing Files
* Working with File Paths
* Context Managers

1. **Other topics**

* Programming Paradigms in Python.
* Method Resolution Order (MRO).
* Memory Allocation.
* Memory Management Mechanisms.
* Memory Leak in Python.

1. **Object-Oriented Programming (OOP)**

* Classes and Objects
  + constructor
  + Deconstructor
  + self parameter
* Instance Variables
* Methods
  + Static method
  + Instance method
  + Class method
* Inheritance
  + super()
  + Single Inheritance
  + Multiple Inheritance
* Polymorphism
* Encapsulation
* Abstraction
* Magic Methods
  + init, str, repr, etc.

1. **QA**

* role of init.py in package?
* Python is Intrepreter language ?
* Deffrence between oops and pops ?
* Mutable vs immutable?
* Shaloow coppy, deep coppy ?
* What is .pyc file ?
* Generate random number between 1 and 100 using lambda function?
* Deffrent between “is” and “==”?
* What is itrable ?
* What is primitive and non primitive ?

staticmethod vs classmethod

\_\_name\_\_

Regex

Iterator

Garbage collection

Pickling

Multy thrud language

Meta class

Multy processing

Memmary management

Referance conting

split

# Others

### SASS

1. @import "../node\_modules/bootstrap/scss/bootstrap";
2. @use & @forward

### REST API

1. it’s about communication
2. RESTful
3. pros
   1. simple & standardised
   2. scalable & stateless
   3. high performance due to cachings
4. **Request**
   1. General (start line)
      1. method/target/version
   2. operation: get, post, put, delete
   3. endpoint
   4. header
      1. API key
      2. authentication data
   5. body/ parameter
5. **Response**
   1. General (start line)
      1. version/statuscode/statustext
   2. header
      1. content type
   3. body
      1. requested resource
6. **HTTP Methods**
   1. GET
   2. POST
   3. PUT
   4. DELETE
7. Idempotent
8. Headers
9. Status code
   1. 1xx: Informational
   2. 2xx: Success
      1. 200 - Success
      2. 201 - Success and created
   3. 3xx: Redirect
      1. 301: moved to new URL
      2. 304: not changed
   4. 4xx: Client Error
      1. 401: Unauthorised
      2. 402: 402 Payment Required
      3. 403: Forbidden
      4. 404: page not found
   5. 5xx: Server Error
10. MIME type
11. HTTP v2
12. TCP and IP

### CI CD (git)

### JSDoc

1. /\*\*  
   \* function description  
   \* @param {string} description  
   \*/
2. Params
3. Returns

### Sequelize

### Testin

### Swagger